Lab# 10 THE 16-BIT ALU IMPLEMENTATION

Instructor: I Putu Danu Raharja.

Objectives:
Learn to implement a simple 16-bit single-stage non-pipeline ALU circuits.

Method:

Complete the circuit for a 16-bit Barrel Shifter and verify it's operation using the simulator.

Preparation:

File To Use:

10.1 OVERVIEW:

The MIPS ALU (arithmetic and logic unit) performs all of the core computations dictated by the assembly language. You should have already designed a 1-bit full adder and subtractor circuit in Lab 9. Logisim comes with libraries containing basic gates, memory chips, multiplexors and decoders, and other simple components. You may use any of these basic components in your designs, but you may not use the Logisim arithmetic library anywhere in your design.

The ALU design below takes three inputs: first argument (A16-bits), second argument (B 16-bits) and the ALUCtrl (4-bits). Also, it has 3 outputs: output (C 16-bits), Overflow Flag (OF 1-bit) and Zero-Flag (ZF 1-bit). Based on the ALUCtrl value, the following functions will be implemented:

ALUCtrl	C	Function Name
000x	C = B << sa	shift left logical
001x	C = A + B	add
0100	C = B >> sa	shift right logical
0101	C = B >>> sa	shift right arithmetic
011x	C = A – B	subtract
100x	C = A & B	and
101x	C = A | B	or
110x	C = A ^ B	xor
111x	C = ~(A | B)	nor

Begin by implementing the following circuits (numbers in brackets give the number of bits in each input/output).

A. Adder and Subtractor:

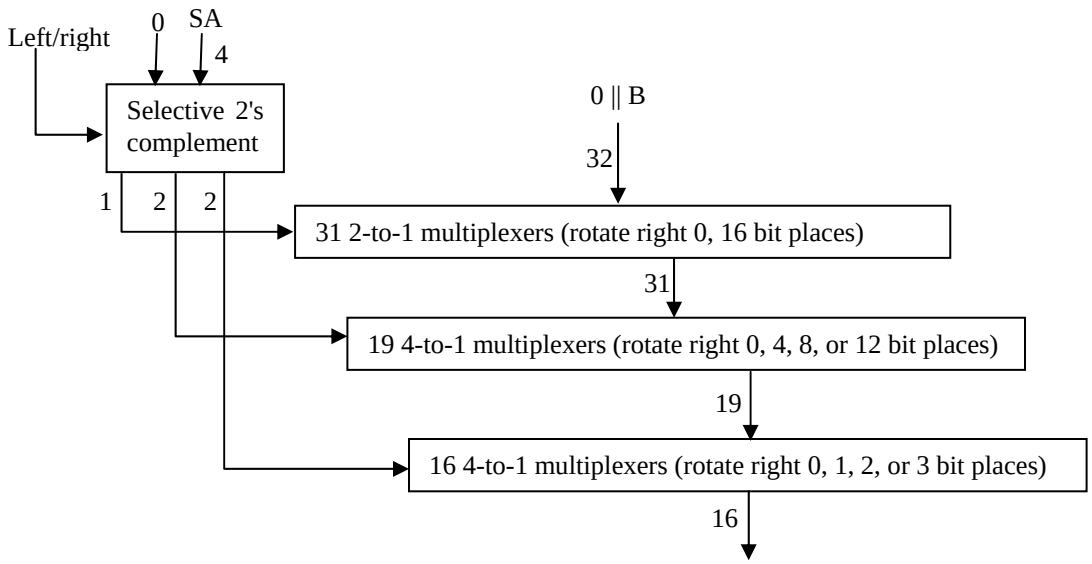
Input: A[16], B[16], cin, M. Ouput: C[16], cout, OF. If M==0 then C = A + B + cin, The output C is computed by adding A, B, and cin. Any remaining carry bit is output on cout. If M==1 then C = A – B. In this case, cin=M=1. and input B will be negated.

Hint: Use sub-components to make wiring easier, by building a 1-bit adder, then 8-bit adder, then 16-bit. The overflow flag may be detected by comparing cout and the carry from bit 14.

B. Shifter:

We can use a barrel shifter to implement logical right or logical left shift operation from 0 to 15 positions. The data input is 16-bit operand B, and the output is 16-bit result C. Left/right a control signal decoded from OPCODE, select a left or right shift. The shift amount field SA(3:0) specifies the number of bit positions to shift the data input and takes on values from 0 through 15. A logical shift of P bit places involves inserting P zeros into the result. In order to provide these zeros and simplify the design of the shifter, we will perform both the left and right shift by using a *right rotate* circuit. The input to this rotate will be the input B with 16 zeros concatenated to its left. A right shift is performed by rotating the input P places to the right; a left shift is performed by rotating 32 – P places to the right. This number of places can be obtained by taking the 2's complement of the 5-bit value of 0 || SA.

The 31 different rotates can be obtained by using three levels of 4-to-1 multiplexers, as shown in the following figure. The first level shifts by 0 or 16 places, the second level by 0, 4, 8, or 12 places; and the third level by 0, 1, 2, or 3 places. Due to the presence of 16 zeros in the 32-bit input, fewer than 32 multiplexers can be used in each level. A level requires the number of multiplexers to be 16 plus the total number of places its output can be shifted by subsequent levels. The output of the first level can be shifted at most 12 + 3 = 15 places to the right. Thus this level requires 16 + 15 = 31 multiplexers. The output of the second level can be shifted at most 3 places, giving 16 + 3 = 19 multiplexers. The final level cannot be shifted further and so needs just 16 multiplexers.



Note the difference between logical right shift (which fills with zero bits), and arithmetic right shift (which fills the empty bits with the copies of the sign bit of B).

10.2 LAB EXERCISE

Implement the above design using Logisim.